PVFS2 and Parallel I/O on BG/L

Rob Ross

Mathematics and Computer Science Division

Argonne National Laboratory

Special acknowledgements

- Rob Latham – did most of the work to get PVFS2 up and running
- Susan Coghlan – provided all the access we needed, made everything easy for us, and clicked the mouse at the right time (even from SLC!)
- Kazutomo Yoshii – figured out how to get things built for the IO nodes at Argonne
- LLNL group (Robin, Ira, others) – provided us with a great start for building IO node kernels
- IBM – provided source to key components and insight into system components that made this possible

Outline

- PVFS2 introduction and background
  - What it is, who it is, and why it's interesting for BG/L
- Base functionality for PVFS2 on BG/L
  - What is working, preliminary performance numbers
- Beyond the baseline
  - Pursuing higher I/O performance
  - Research in MPI-IO
- Wrap up

The PVFS2 Parallel File System

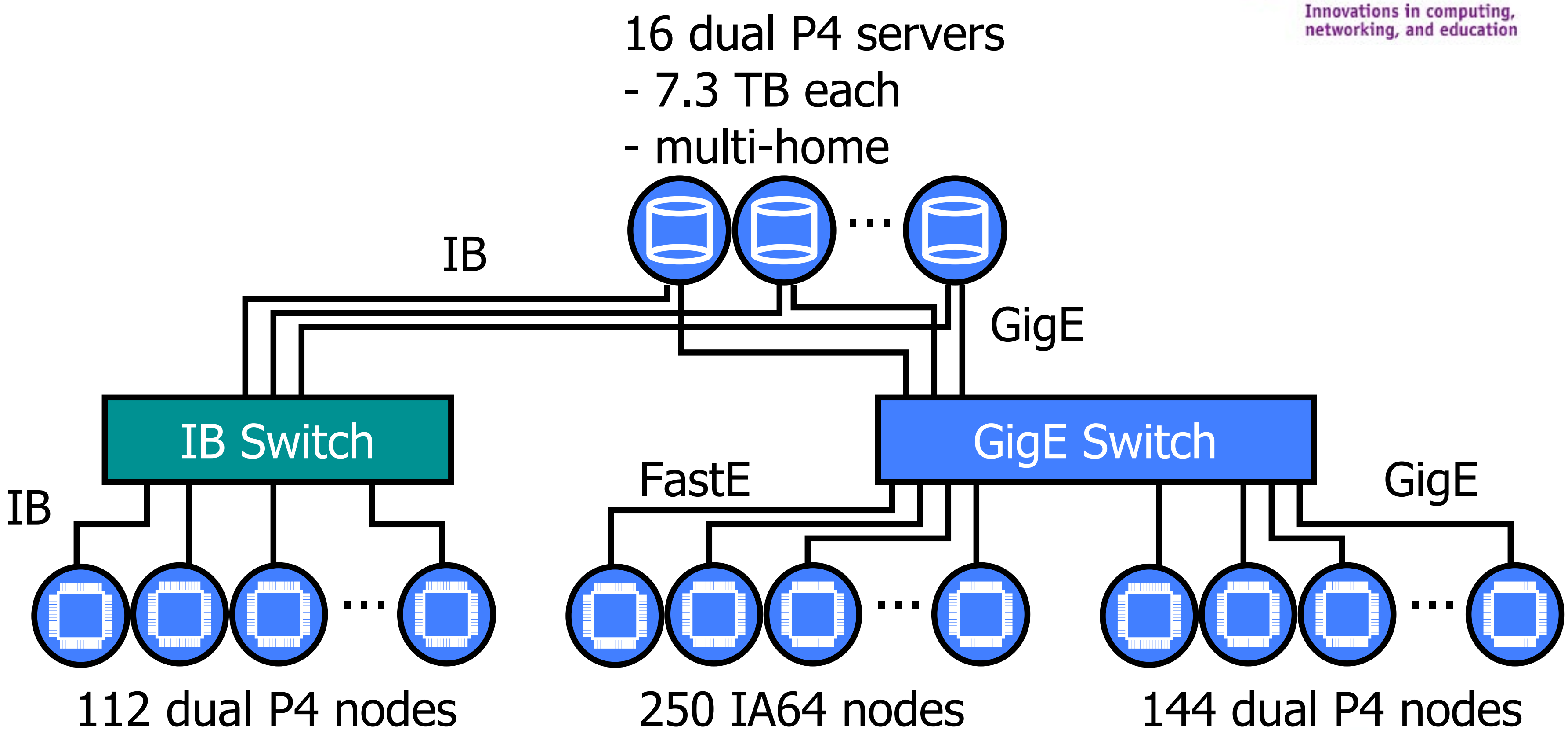
- <u>Parallel</u> file system
  - Distributed data and metadata
  - Tuned for performance and concurrency
- Production ready
  - In use at ANL, OSC, Univ. of Utah CHPC, others
- Open source and open development
  - LGPL license on all but kernel module, GPL on kernel module
  - Current CVS is anonymously accessible
  - Mailing lists where developers can track and initiate discussions
- Community research vehicle
  - Heterogeneous system support
  - Predominantly user-space code
  - Rapid porting via network and storage abstractions
  - Many labs and universities extend or modify PVFS2 to explore new ideas

Who is PVFS2?

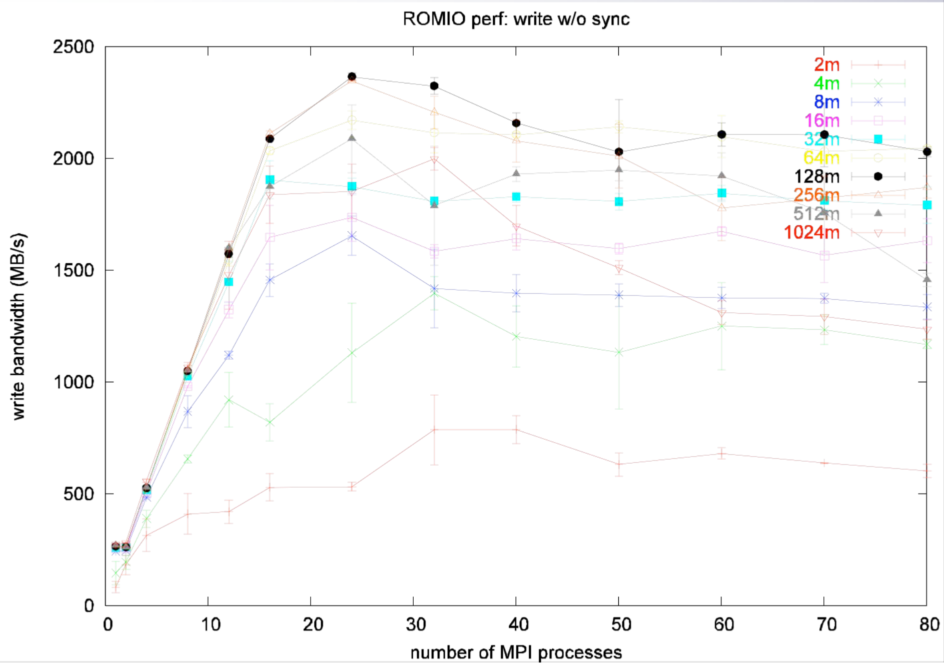
- PVFS2 is an open, collaborative effort
- Core development
  - Argonne National Laboratory
    - Ross, Latham, Gropp, Thakur
    - Supported by DOE Office of Science
  - Clemson University
    - Ligon, Settlemyer
  - Ohio Supercomputer Center
    - Wyckoff, Baer
- Collaborators
  - Northwestern University
    - Choudhary, Ching
  - Ohio State University
    - Panda, Yu
  - Penn State University
    - Sivasubramaniam, Kandemir, Vilayannur

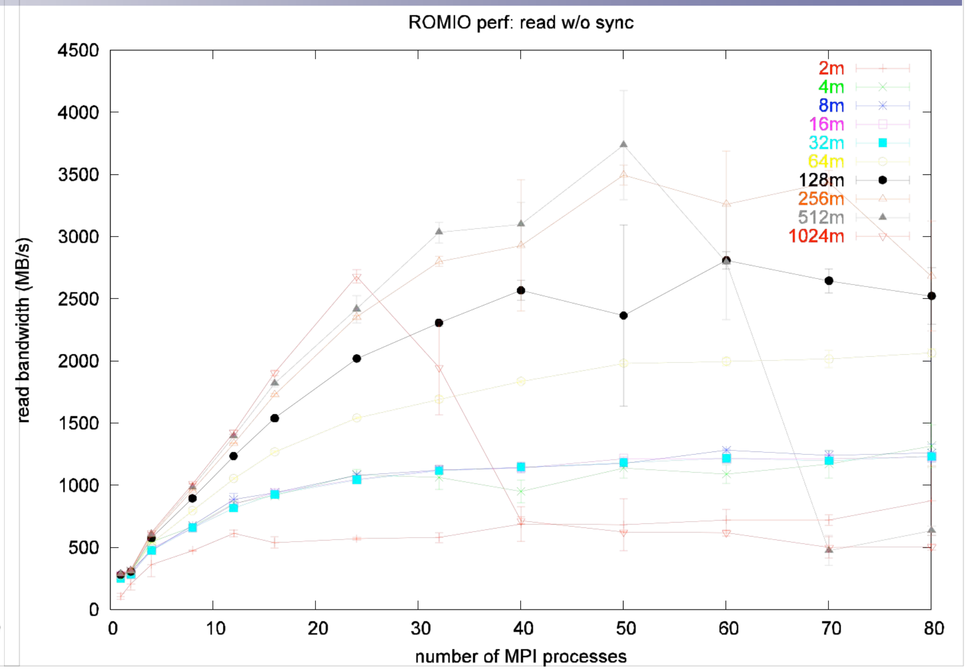
PVFS2 at OSC

- 506 total clients
- 116.8 TByte file system



OSC cluster performance

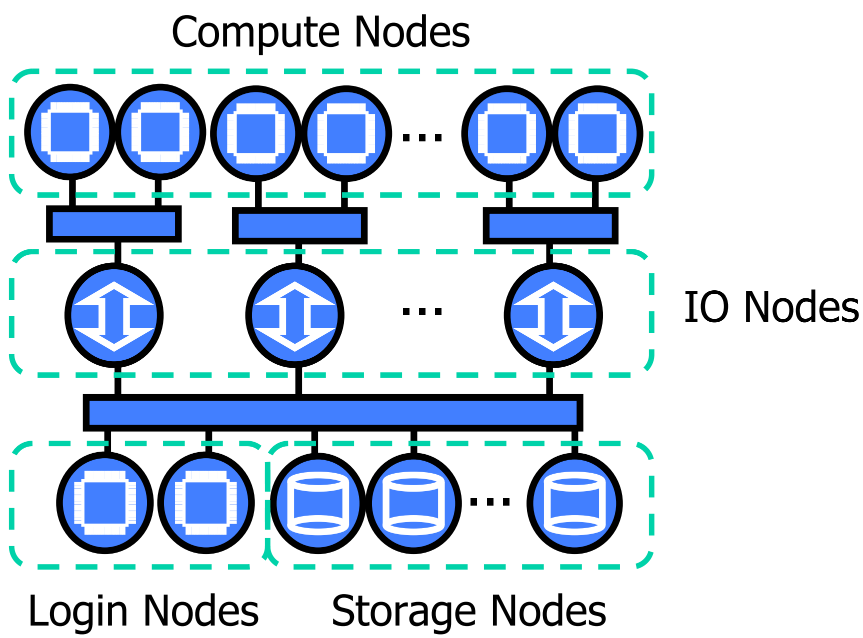




- Data sizes are per-client
- Achieving ~2.8GB/sec write, ~3.8GB/sec read
  - No network optimization (memory registration or pipelining)

View of I/O on BG/L

- Storage nodes
  - Local access to disks
  - GigE connections to login and IO nodes
- Login nodes
  - Interactive machines
  - Place where data staging will occur
- IO nodes
  - Aggregators for compute node I/O
    - 1:8 to 1:64 ratio of IO nodes to compute nodes
  - Tree connection to compute nodes
- Compute nodes
  - Source/sink of runtime I/O

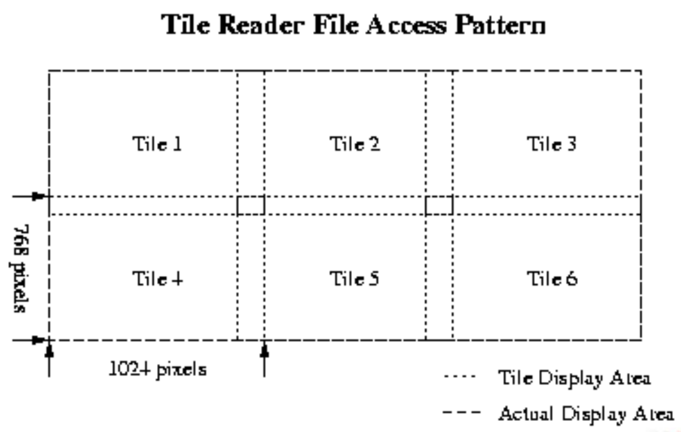


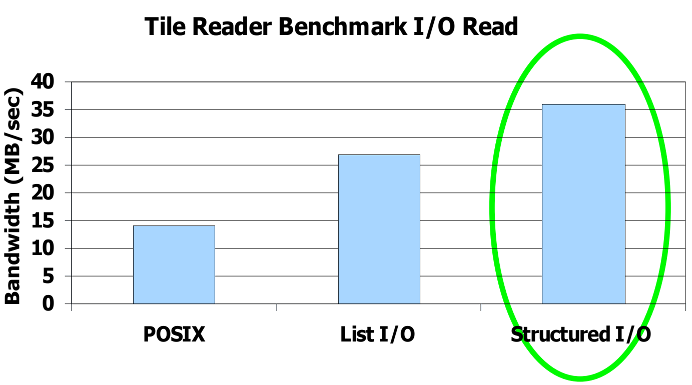
Why put PVFS2 on BG/L?

- It's fun ☺
- It provides another data point for I/O performance
- Most importantly, PVFS2 addresses three key scalability problems for parallel file systems:
  - I/O performance (especially for noncontiguous data)
  - Metadata performance (in particular open/close)
  - Failure tolerance
- Because of these advantages, we believe that PVFS2 has the best chance of extracting the highest possible I/O performance from BG/L

Scaling effective I/O rates

- POSIX I/O APIs aren't descriptive enough
  - Don't allow us to generally describe noncontiguous regions in both memory and file
- POSIX consistency semantics are too great a burden
  - Require too much additional communication and synchronization, not really required by many HPC applications
  - Will never reach peak I/O with POSIX at scale, only penalize the stubborn apps
  - Use more relaxed semantics at the FS layer as the default, build on top of that





ARGONNE NATIONAL LABORATORY

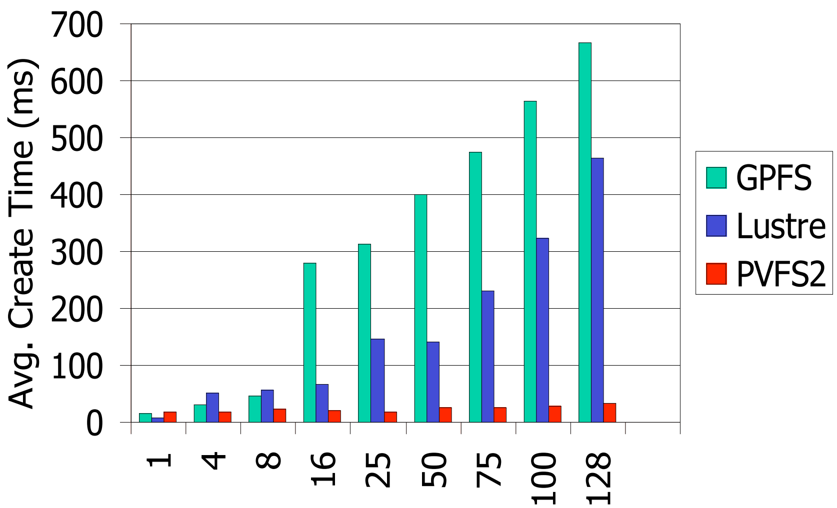
PVFS2 PARALLEL VIRTUAL FILE SYSTEM

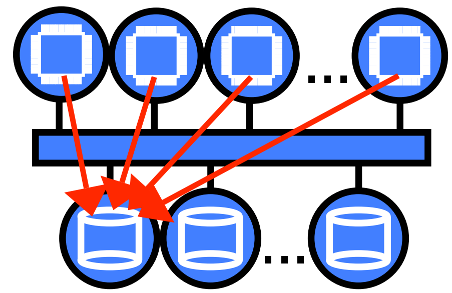
Office of Science U.S. DEPARTMENT OF ENERGY

Scaling metadata operations

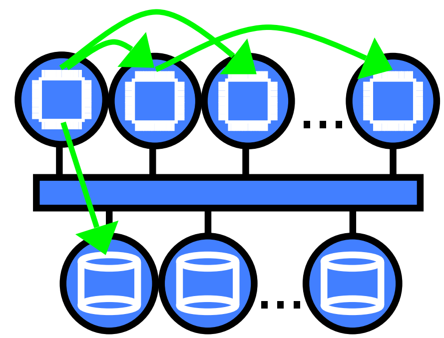
- POSIX API hinders scalability here too
  - POSIX open/close access model imposes constraints on how we implement MPI-IO operations like MPI_File_open
  - Similar issues with fsync and other operations

MPI File Create Performance (small is good)





POSIX file model forces all processes to open a file, causing system call storm.



Handle-based model uses a single FS lookup followed by broadcast of handle (implemented in ROMIO/PVFS2).

ARGONNE NATIONAL LABORATORY

PVFS2 PARALLEL VIRTUAL FILE SYSTEM

Office of Science U.S. DEPARTMENT OF ENERGY

Tolerating client failures

- Client failures are likely to be common with high node counts
  - 99.99% up indicates ~6 nodes down at any time on a 64K node system
  - 99.9% up indicates ~65 down at any time on same
- Unlike other options, PVFS2 uses a **stateless I/O model**
  - No locking system to add complications
  - No other shared data stored necessary for correct operation (no tracking of open files, etc.)
- Client failures can be ignored completely by servers and other clients
  - As opposed to locking systems, where locks and dirty blocks must be recovered!
- Server restarts are easily handled as well

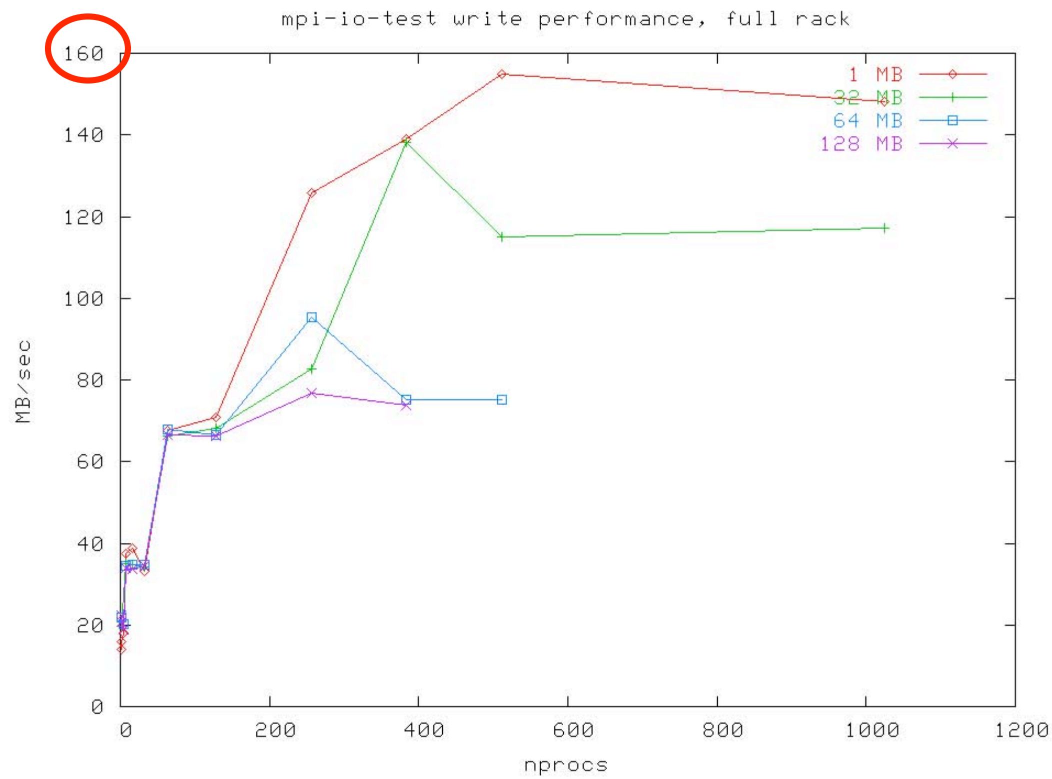
First steps in running PVFS2 on BG/L

- **Goal: Enable data staging and runtime I/O to a PVFS2 file system**
  - Run PVFS2 servers on storage nodes
    - dual Xeon nodes running SLES Linux and 2.6.5 kernel
  - Mount PVFS2 file system on login nodes
    - PowerPC 970 nodes running SLES Linux and 2.6.5 kernel
  - Mount PVFS2 file system on IO nodes
    - BG/L PowerPC nodes running 2.4.19 kernel (no longer MontaVista)
- **This only took two weeks to accomplish!**
  - Mostly learning/creating build environment
  - Minimal patching to PVFS2 (all in CVS)
  - 12 PVFS2 servers providing a single coherent file system

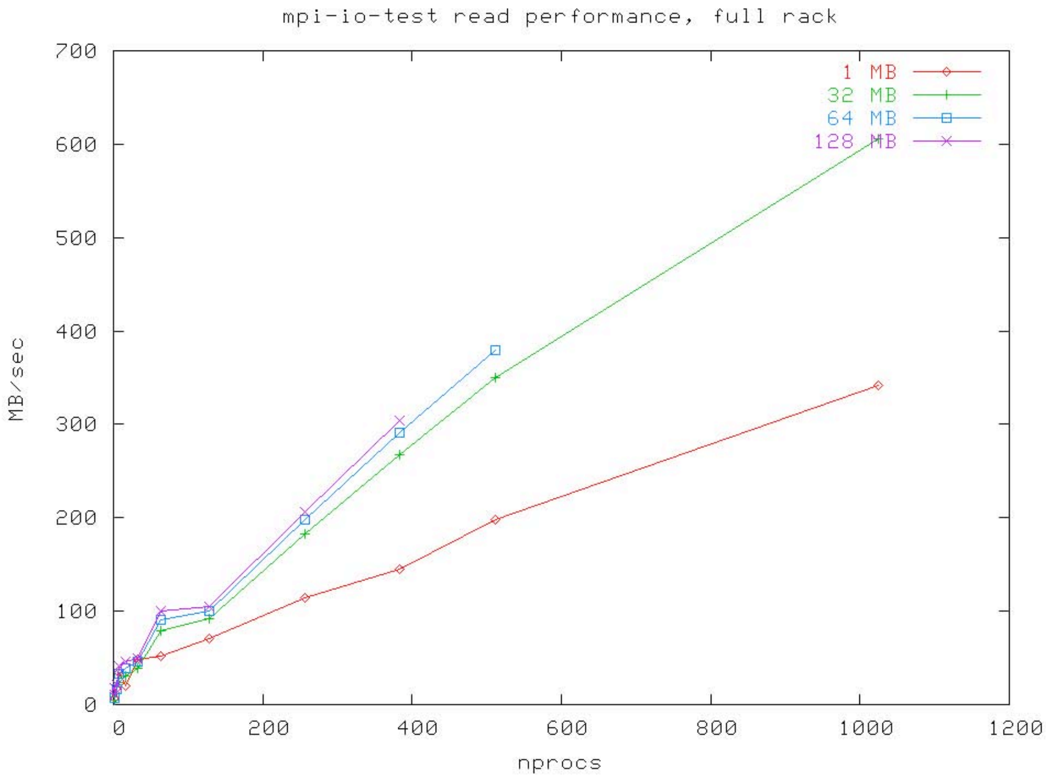
    (Assuming 900mbit/sec network to each, peak of ~1.3GB/sec raw BW)

Write performance (the bad news)

- Simple pattern:
  - Single file
  - Independent MPI-IO
  - One big access each
- We have more work to do here!
- ciod is breaking accesses into 95520 byte blocks
  - Understand why better now (Mike's talk)
  - Try tuning I/O message size
    - **What's the variable?**
  - Check TCP buffer sizes and turn on jumbo frames (Chris's talk)
  - Why does strace'ing the ciod kill our machine sometimes?
  - "Happy SuperComputing!" to you too ☺



Read performance (the good news)



- Peak of 600MB/sec (44% of raw BW, also no tuning)
  - This is **with** those tiny blocks…

ARGONNE NATIONAL LABORATORY
PVFS2 PARALLEL VIRTUAL FILE SYSTEM
Office of Science U.S. DEPARTMENT OF ENERGY

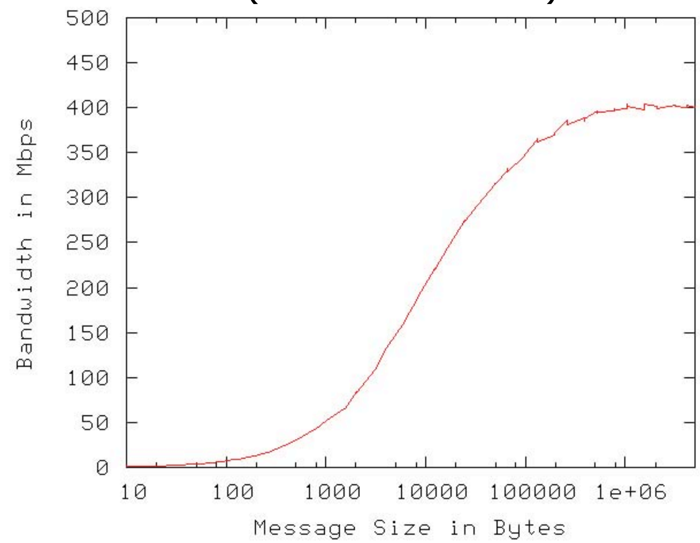
Beyond base functionality

- Our research indicates that the POSIX interface limits I/O scalability
  - Noncontiguous read and write performance
  - Open/close problems
- PVFS2 improvements cannot be seen through the VFS interface
  - BG/L has already broken POSIX, so on the right path…
  - We're still going through the VFS
  - The ciod is using POSIX calls
- **To obtain the highest possible performance we must circumvent (or change!) the VFS**
- Two options:
  - Direct compute node to storage server communication
  - **Retool communication between compute and IO nodes and mechanism IO node uses to access file system**

Direct PVFS2 access from compute nodes

- **Idea: Use PVFS2 client library directly on top of socket call forwarding to bypass IO node mount point**
- BGL PowerPC nodes
  - Special, proprietary kernel
  - Not all system calls are forwarded
- PVFS2 client code will (now) build for compute nodes, but
  - poll() and select() aren't implemented, so we can't run
- Interesting experiment, but not ideal solution…

TCP Bandwidth over Tree
(modified NetPIPE)



Changing the I/O language

- **Really what we'd like to do is change how compute processes talk to the file system**
  - Ideas prototyped in PVFS2 already
  - Allow for efficient noncontiguous I/O
  - Eliminate open() and close() scalability issues
  - More efficiently leverage the tree, IO node, and GigE
- This means changing how compute processes communicate with the IO node
  - Replace or augment existing ciod functionality
  - Map new language to PVFS2, GPFS, Lustre operations
    - **These changes can benefit any underlying file system**

I/O research in BG/L

- **Plan:** Experiment with new MPI-IO algorithms
  - Control of access mapping to IO nodes
  - Caching of data at IO nodes (to what extent possible)
  - New GPFS, Lustre, PVFS2 optimizations
- To do this, we must be able to rebuild ROMIO and link to IBM MPI
- **Next Tasks:**
  - ANL ensures that ROMIO builds cleanly against IBM MPI
  - IBM provides MPI without ROMIO

Wrap up

- In a couple of weeks we were able to get PVFS2 running on BG/L
  - Open source operating systems played a key role
  - Very positive experience!
- IBM developers have been very helpful
  - Will aid greatly in MPI-IO research and tuning for BG/L
- This is turning into an ideal platform for testing and deployment of next-generation I/O systems!
- High level libraries will follow as well
- We could use just a little more source… ☺

ARGONNE NATIONAL LABORATORY PVFS2 PARALLEL VIRTUAL FILE SYSTEM Office of Science U.S. DEPARTMENT OF ENERGY

Additional information on PVFS2

- PVFS2 web site: http://www.pvfs.org/pvfs2
  - Documentation, mailing list archives, and downloads
- PVFS2 mailing lists (see web site)
  - Separate users and developers lists
  - Please use these for general questions and discussion!
- Email
  - Rob Ross <rross@mcs.anl.gov>
  - Rob Latham <robl@mcs.anl.gov>

ARGONNE NATIONAL LABORATORY PVFS2 PARALLEL VIRTUAL FILE SYSTEM Office of Science U.S. DEPARTMENT OF ENERGY